A Strong Partitioning Protocol for VME-based Systems Integration

Mohamed F. Younis **Jeffrey X. Zhou**

AlliedSignal Inc.
Advanced Systems Technology Group
9140 Old Annapolis Road
Columbia, MD 21045
Younis|Zhou@batc.allied.com

Abstract

This paper describes a new technology that can provide a global VME memory management in supporting the need of strong partitioning among multi-processor applications on the VME backplane. The technology allows each VME board to maintain a fault containment region on board level and prohibit fault propagation through the bus. The described techniques do not require any modification in the standard and the existing boards, and consequently, maintains the plug-and-play advantage of the VMEbus hardware products. The approach is to use a message passing mechanism for multiprocessing instead of shared memory. A software layer has been developed to support message-based inter-module communications using the available features provided by any standard VME card and still detect errors and prevent fault propagation.

1. Introduction

Advancements in technology have enabled the avionics industry to develop new design concept, which results in highly integrated software-controlled digital avionics. The new approach, referred to as Integrated Modular Avionics (IMA), introduces methods which can achieve high levels of reusability and cost effectiveness compared to earlier implementations of avionics [1]. The IMA approach encourages partitioning and using standardized building blocks in building environmental and functional components of avionics. Strong functional partitioning facilitates integration, validation and FAA certification. Following the IMA guidelines, the cost of both development and maintenance is expected to decrease because of large quantity production of the building blocks, lower levels of spares, and reduced certification costs.

The goal of this work is to develop a fault tolerant architecture that encourages functional partitioning and use of commercial-off-the-shelf (COTS) technology in implementing integrated control systems for both commercial and military aircraft. Integration of electronic controls is highly demanded by OEMs to reduce weight and maintenance costs. In addition, the integrated approach allows sharing of common resources such as power supplies, cooling system, data bus, etc. The use of a fault tolerant architecture for the integrated control is a necessity to minimize the risk of loosing multiple control functions due to some hardware or software failure in the unified environment.

The airborne backplane bus is one of the most important components in integrated modular avionics. While much backplane bus designs have been proposed, only a few are actually used. Selecting the backplane bus is affected by many design and engineering factors, such as performance, reliability, and fault-tolerance. Although, such issues are very important to ensure certain level of safety of commercial jet aircraft and high availability of military aircraft, the cost of the bus and associated line replaceable modules is a major concern.

Most of the currently available dependable backplane bus systems are expensive and supplied by very few vendors, such as ARINC 659[2]. It is clear that there is a need for an affordable bus system that provides the required levels of dependability for an integrated utility control and complies with the IMA design methodology. This paper shows how to enhance functionality and overcome inefficiencies in commonly used and widely manufactured low-cost buses to make them suitable for an integrated control. Since the trend in implementing today's real-time embedded applications is toward the use of commercial-off-the-shelf open architecture to reduce costs and facilitate system integration, the VMEbus system [3] is a prime candidate because it is both rigorously defined and widely supported. In addition, there is an expanding selection of VMEbus boards and vendors that guarantee competitive prices and continuous support. Moreover, the VMEbus offers an open architecture that facilitates the integration of multiple vendors' boards. Such features make the VMEbus an attractive choice for integrated utility control.

However, the VMEbus standard does not impose strong functional partitioning and allows fault propagation from one board to another, as we discuss in *Section* 3. Such weakness limits the use of the VMEbus in an integrated control system. This paper presents techniques for strong partitioning of multi-processor applications that maintain fault containment on the VMEbus. The suggested techniques do not require any modification in the standard and the existing boards, and consequently maintain the cost advantage of the VMEbus hardware products. In addition, a fault tolerant architecture is proposed for an integrated control system. The architecture includes commercial-off-the-shelf hardware and software components and other AlliedSignal- developed products to provide a reliable and cost-effective integrated control platform.

2. An Overview of the VMEbus

The VMEbus design is highly influenced by the development of the Motorola MC68000 microprocessor, although nowadays VME boards are available for many other processors. The VMEbus allows multi-processing, expandability and adaptability by many design and processors. It handles data transfer rates at speeds in excess of 40 Mbytes/sec using parallel data transfer.

The VMEbus is asynchronous and non-multiplexed. Because it is asynchronous no clocks are used to coordinate data transfer. The system clock is 16Mhz and has no relation to the other bus activities. Typical uses of it include bus timers, memory refresh circuits, serial I/O time bases and synchronous state machines. Data is passed between modules using interlocked handshaking signals where cycle speed is set by the lowest module participating in the cycle. Using asynchronous protocol, in the VMEbus, provides reasonable compatibility to integrate products from various vendors.

Master-slave architecture is used in the VMEbus. Modules can be designed to act as master, slave or both, although some modules can be configurable. Before a master can transfer data it must first acquire the bus using a central arbiter. This arbiter is part of a module called the system controller. Its function is to determine which master gets the next access to the bus. The bus arbiter grants the bus according to prioritized requests handling or simply using round robin. Bus grant signals are propagated back to the requester through a daisy chain over all modules between the requester and the system controller. Four levels of arbitration are provided, each with its own daisy chain. Every module has a set of jumpers to select the arbitration level. The bus arbiter may prioritize requests according to its arbitration level.

All interrupt service routines are user defined. The daisy chain again is used to propagate the acknowledgement signal from module to module until it reaches the interrupter. The interrupter then places a STATUS/ID on the bus to notify the handler of which card initiates the interrupt.

Due to the daisy chaining used for bus mastership requests and interrupt, modules are position-sensitive. The first slot is reserved for the system controller. The closer the module to the system controller, the higher the privilege of getting the data bus and the higher the priority of this modules interrupts.

The VMEbus provides support for multiprocessing using shared memory. To avoid inconsistency while updating shared memory, read-modify-write bus cycles are to be used. The read-modify-write cycle allows updating shared memory as an atomic transaction and prevents race conditions.

Although the VMEbus does provide reasonable compatibility to integrate products from various vendors, fast parallel data transfer, and a wide support by many manufactures, fault-tolerance in VMEbus based systems is very limited. The next section discusses fault detection mechanisms in the VMEbus and elaborates weaknesses in fault containment and fault-tolerance.

3. Fault-Tolerance Challenges in the VMEbus

The VMEbus relies on modules for detecting and reporting faults on a specific failure control line. VMEbus modules are expected to have on-board firmware diagnostics to detect faults. In response, the system controller polls every module by reading its status register to identify the faulty module. Generally, the build-in-test and transmission time-out provides limited fault coverage for only permanent faults.

The VMEbus master (sender) monitors the time for data transfer. If the receiver does not acknowledge the message, the master times out data transfer and retransmit. The bus does not provide error detection or correction for the transferred data. There is no redundancy in either the transmission lines or the transferred data on the bus. In addition, the VMEbus system allows a single point of failure by using a centralized mastership control of the bus. The system controller organizes all communications between modules. Moreover, the system controller is responsible for monitoring the failure control line and for tracing the faulty module, which derived the failure line. Faults in the system controller can bring down the whole system.

The shared memory model used by the VMEbus for multiprocessing makes the modules tightly coupled. In the absence of message verification, faults can propagate from one module to the others. Errors cannot be contained within the faulty module and can jeopardize the behavior of the whole system.

The daisy chain reliability is questionable. Each module either accepts the bus grant or passes it over to the next module. Failure of one module may break the chain and affect other modules. Breaking the chain can prevent other modules from getting mastership of the bus and from reacting to interrupts. Communication between modules can be highly affected and the whole system can break down.

From the former discussion we can conclude that the VMEbus needs enhancements to strengthen its fault-tolerance capabilities, specifically in containing errors and recovery from failure. The following issues need to be addressed in order to improve containment of faults in a VMEbus system:

1. validating the inter-module data transfer.
2. propagating faults through the use of shared memory.
3. breaking the daisy chain because of a module failure.

In addition, fault recovery mechanisms need to be added to:

1. prevent having the system controller as a single point of failure.
2. allow redundancy for the system critical functions.

The next sections provide a discussion of our approach to address these issues starting with techniques for fault containment.

4. A Fault tolerant VMEbus

Because low cost is an important feature of the VMEbus, enhancing the fault containment capabilities should avoid changing the design and the layout of the currently available cards. Changing the design a VME card will not only require reengineering and revalidation which increases the manufacturing cost, but also will again limit the number of vendors who agrees to do the modifications. Thus, the suggested approach should be constrained by preserving the current hardware design of the cards as much as possible.

As illustrated in the previous section, both fault containment and failure recovery features need to be improved. In the following subsections, we present our approach to enhance these features.

4.1 Fault Containment Techniques

As illustrated in the previous section, fault containment on the VMEbus needs to be added. In the following subsections, our approach is discussed.

4.1.1 Inter-module Data Transfer

The VMEbus features parallel data transfer between modules. There are no error detection or correction bits associated with the transmitted data. Adding such bits will significantly affect the VME card design and, therefore, is not an option. As an alternate approach, an error detection code, e.g. cyclic redundancy check, can be appended to the end of the data. The message transmission module within the operating system kernel can generate the code. Although the software-generated error detection code is less efficient than the hardware-based implementation, no card redesign is necessary using the software approach. For higher dependability, an error correction code can be appended. Because that error detection/correction code will reduce the efficiency of the data transfer on the bus and consequently the performance, it may be possible through the kernel to dynamically select either to append error detection or error correction code according to the length of the transmitted data. The receiver module should validate the data using the error detection/correction code before committing that received data.

Using such information redundancy within the transferred data fits the multiprocessing scheme proposed in the next subsection.

4.1.2 Multiprocessing

Strong partitioning of modules is one the most important IMA requirements which the VMEbus lacks. Multiprocessing in the VMEbus uses a shared memory mechanism that allows faults in one module to cause errors in other non-faulty modules by writing to their memories. In our approach, we used a message-passing mechanism instead. The challenge is to support message-based inter-module communications using the available features provided by the VME cards and still detect errors and prevent fault propagation.

To support messages, a buffer is to be declared and dedicated only for messages. A message buffer is the only globally visible memory of a module. Modules are not allowed to access the memory of other modules other than their message buffers. In addition, access to a message buffer is restricted to read-only for modules that do not own that buffer. No card is supposed to write to the memory of other cards. If a master wants to send data to a slave, it simply writes a message for the slave into the master's message buffer. The slave reads that message from the master memory and reacts to it.

A specific message format can be imposed that contains the sender ID, receiver ID, error detection or correction code, and a message unique ID. The sender should perform error detection and correction encoding. The slave will check the contents of the message before reacting to it. The receiver can detect addressing errors in the message by verifying the sender ID and receiver ID. In addition, transmission errors can be detected or recovered using the information redundancy in the form of the error detection or correction code in the message.

Synchronization can be achieved either by polling the message buffer of the sender for the required message, or by using the address monitoring feature provided by the VMEbus to interrupt the receiver as soon as a message is being written by the sender in the designated address. The message ID can be useful to overcome race conditions if the receiver tried to read the message before it is ready which may be is possible if the VMEbus has a higher priority than the local bus. The message buffer can be partitioned for various cards and slaves can expect a unique location for their messages. The adopted application execution-synchronization mechanism is a designer decision.

Using this technique, errors in the sender can be isolated and prevented from propagation to the receiver. A fault in the sender may affect the receiver only through the message. As explained earlier, no write permission will be granted for a card to the memory of others. Errors in the message can be either in data, sender ID, receiver ID, Message ID, or message format. The receiver should be able to detect errors in the message body by validating the message format, error detection code, sender ID and the receiver ID. The message ID can be checked to guarantee the right message sequence. Any error in the message detected by the receiver will invalidate the entire message and a recovery action will be taken. An addressing fault in the receiver that may get it to read from the wrong card or the wrong address within the right card will affect the message format and the sender ID. Furthermore, the mapping of message buffers of cards in the global address space of the VME system should be widely distributed so that an addressing error can not change a valid global address into another valid address. Maintaining a suitable hamming distance can guard the system against permanent or transient stuck failure of one or more address bits. Thus, the system will be functionally partitioned. Faults can be within the faulty module and will not affect other modules.

4.2 VMEbus failure

The VMEbus relies on a centralized control for arbitrating the bus mastership among various cards connected to the bus. The centralized control is vulnerable to a single point of failure that can bring down the whole system. In addition, a daisy chain is used in the VME backplane to propagate bus grant and interrupt acknowledgment signal. A failure in one module may break the chain and affect the following modules in the chain preventing them from communicating over the VMEbus.

One approach to tolerate failure of the VMEbus is to provide an alternate path for inter-module communication as shown in *Figure 1.* Many vendors include a secondary bus on their VME boards such as the *VME Subsystem Bus* (VSB) [5] and the *RACEway Interlink* [6] to act as a local subsystem extension bus. Both the VSB and the RACEway interlink allow the developer to reduce the VMEbus traffic by using the secondary bus for some DMA and I/O operations. Using either the VSB or the RACEway interlink to tolerate failures in VMEbus requires no modifications in the VME boards and allows the use of cost-effective off-the-shelf boards to develop avionics and other mission critical applications complying with the IMA specifications.

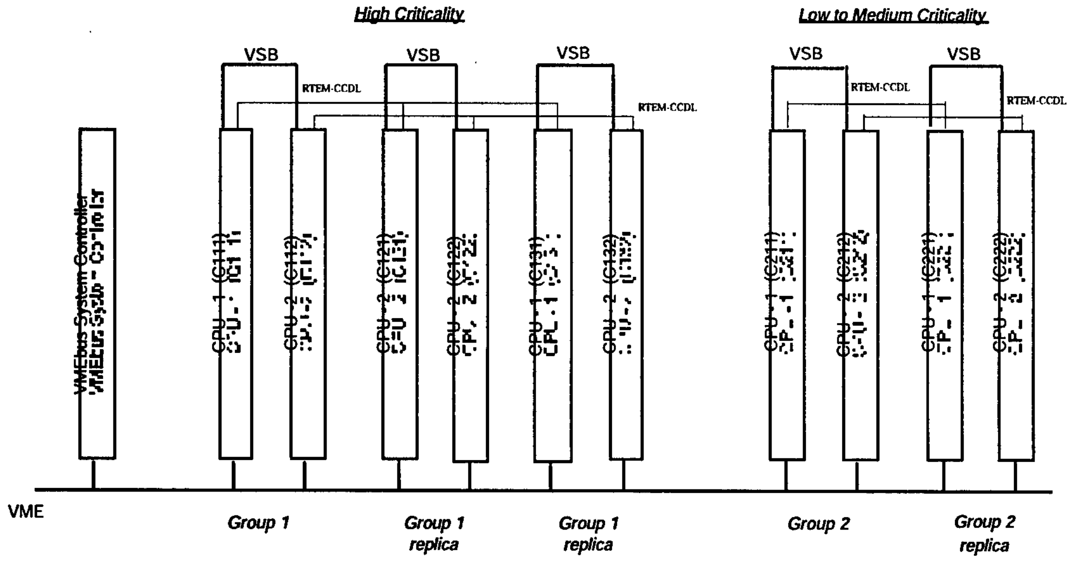

Figure 1: A Fault tolerant VME-based architecture using VSB secondary buses

Another approach for tolerance of VMEbus failure is to use separate redundant VME backplanes that fail independently, as shown in *Figure 2.* The redundant backplanes host modules capable of performing the same functions on the primary backplane.

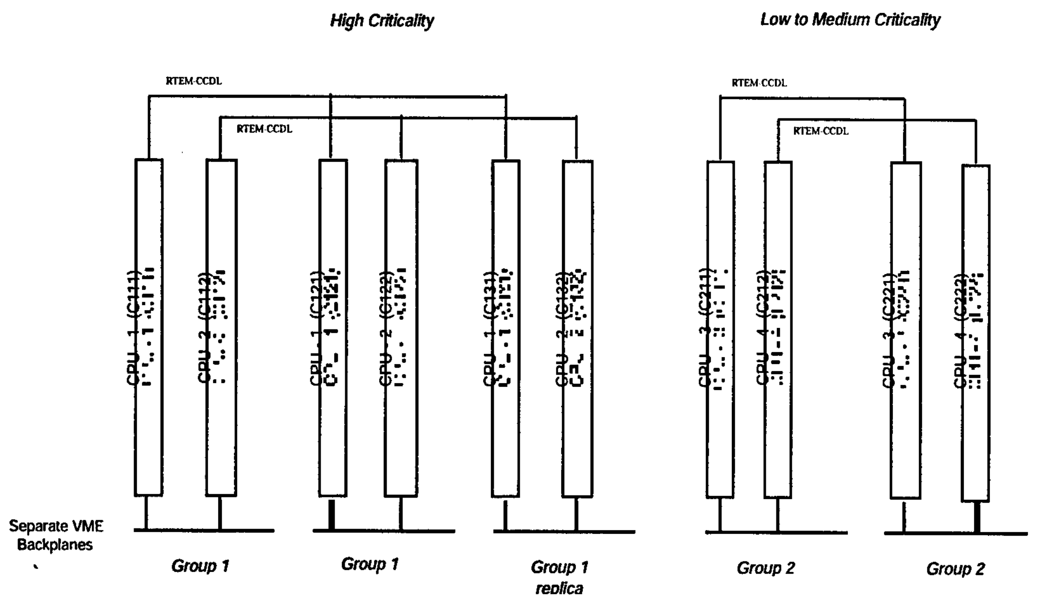

Figure 2: A Fault tolerant VME-based architecture using Separate VME backplanes

Although using a secondary bus or a redundant VME backplane can tolerate a VMEbus failure, it is necessary to allow replication of critical system components and manage such redundancy. In the next subsection, we discuss an approach to managing redundancy. We propose the use of Redundancy Management System (RMS) [7], a fault tolerant executive developed by AlliedSignal, for detecting and masking errors and for redundancy management.

4.3 Redundancy Management

The use of redundancy is very common in fault tolerant systems. Spare units are usually included that can be either passive or active during normal operation. Redundancy management is required to coordinate the interaction between the primary and spare units. Managing redundancy typically needs to be considered during the design and the development of the application.

One or more backup unit is used in active redundancy. A backup will be activated in case of primary unit failure. The backup needs to be updated frequently with changes in the primary internal state (processor execution state and program store). When activated, the backup will resume execution from the last update (checkpoint). The more the frequency of sending of checkpoints to the backup, the shorter the time needed for recovery. However, checkpointing increases the load on the primary unit and may affect its performance during normal operation. Active replication on the VMEbus does not require multiple backplanes. The application software may need careful design to accommodate checkpointing without dramatic effects on the performance. In addition, checkpoints needs to be safely scheduled so that it would not introduce contention on the bus and delay other essential traffic. Another challenge in active replication is related to fault coverage. The activation of a backup depends heavily of the detection of an erroneous condition within the primary processor. The fault coverage within the boards needs to be comprehensive which is hard to guarantee while using commercial-of-the-shelf boards.

Using passive redundancy, the spare units perform the same function performed by the primary unit. The output of all units is subject to voting to tolerate erroneous output from some faulty units. One way of applying this approach to the VMEbus requires the use of multiple backplanes, as the case in the vehicle management computer of the X-33 (experimental prototype for the single step to orbit space shuttle). Alternatively, voting can be performed internally within the same backplane using multiple modules, as shown in *Figure* 2. We intend to use the RMS for redundancy management. RMS ensures a synchronized and consistent execution for a module and its replicas. In addition, it is possible to support different levels of criticality by the number of replicas. For example, using triple redundancy, it is possible to mask permanent, intermitted and transient faults by RMS. For lower criticality functions, a dual redundancy can be used while RMS will provide error checking and will allow fail save operation. In addition, RMS will provide a homogenous error detection capability and fault coverage for all boards in the system regardless their vendor.

5. Fault-Tolerant Architecture

We propose the fault-tolerant architecture shown in *Figure* 2 for an integrated control system. Modules that need to communicate with each other are grouped together, `Group1` and `Group2` in *Figure* 2. Groups are replicated by providing similar modules running the same programs. RMS is used for redundancy management and error detection for the replicated groups. RMS consists of two major components; the Fault- Tolerance Executive (FTE) implemented in software, and the Cross Channel Data Link (CCDL) implemented as a daughter (mezzanine) board.

The VMEbus standard raises important issues in the location of replicated modules. As mentioned in *Section* 2, the priority of bus mastership and interrupt will be granted to the left-most module when multiple requests are made on the same arbitration or interrupt request line. Thus, a replica of a module needs to be assigned the same arbitration and interrupt request line of that module. This will allow predicting the bus schedule. In addition, groups of the highest criticality needs to be inserted in the left-most slots of the VME backplane, so as not to be affected by a failure of a low criticality group which breaks the daisy chain of the VMEbus. The message-passing inter-processor communication mechanism, presented in *Section* 4, will guarantee fault containment for modules within the same group using the VME bus.

The use of RMS in this architecture makes it possible to support different levels of criticality within the same integrated platform. Using dual redundancy with RTEM allows a fail-safe mode of operation. Increasing the number of replicas to three allows the fault masking, while the use of four replicas enable the handling of Byzantine disagreement between replicas. On the other hand, the use of independent backplanes, it is possible to tolerate a VMEbus bus failure. It should be noted that the fault-tolerance issues related to the power supply of the VME backplane is orthogonal and is beyond the scope of this project.

6. Summary

A proof-of-concept prototype has been built using COTS components as shown in *Figure 3.* The prototype includes three VME backplanes, each of them hosts two PowerPC processor boards. VxWorks, the real-time operating system from WindRiver Systems, is purchased and integrated with the hardware. The strong partitioning inter-processor communication protocol has been implemented and integrated within VxWorks. Initial testing has been performed and currently fault injection experiments are being conducted. In addition, the performance of multi-processing communication using the protocol is being measured. A software design document and a user manual have been prepared for application desingers.

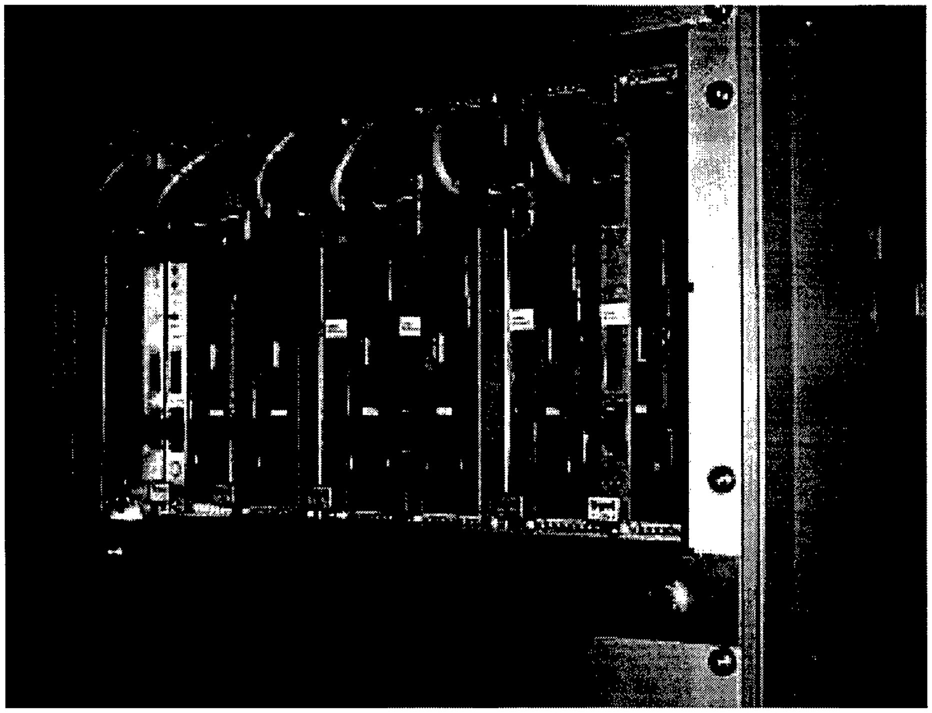

Figure 3: A proof-of-concept demonstration prototype

References

[1] ``Design Guide for Integrated Modular Avionics'', ARINC report 651, Published by Aeronautical Radio Inc., Annapolis, MD, November 1991.
[2] ``Backplane Data Bus'', ARINC Specification 659, Published by Aeronautical Radio Inc., Annapolis, MD, December 1993.
[3] ``IEEE Standard for a Versatile Backplane Bus: VMEbus'', std 1014-1987, Published by The Institute of Electrical and Electronics Engineers, New York, NY, March 1988.
[4] ``Motorola MVME162LX Embedded Controller Programmer's Reference Guide'', Motorola, Inc. Computer Group, Tempe, Arizona.
[5] ``IEEE Standard for a Multiplexed High-performance Bus Structure: VSB (ANSI) Based on IEC 821 Bus, IEC 821-1987'', std 1096-1988, Published by The Institute of Electrical and Electronics Engineers, New York, NY, 1988.
[6] ``RACEWAY Interlink-Data Link and Physical Layers'', ANSI/VITA 5-1994, RACEway Interlink, Published by The VMEbus International Trade Association (VITA), Scottsdale, AZ, 1994.
[7] Jeff Zhou, ``RTEM: A Generic Fault Tolerant Operating System for Real-Time and Mission Critical Applications'', Project Report, AlliedSignal Microelectronics and Technology Center, Columbia, MD, 1992.

PLEASE CHECK THE APPROPRIATE BLOCK BELOW:

-AQ # ____________

☐ ________ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☒ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

☐ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT F:
FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

☐ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE. 6 Nov 1984 (Indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

☐ This document was previously forwarded to DTIC on ________ (date) and the AD number is ____________.

☐ In accordance with provisions of DoD instructions. the document requested is not supplied because:

☐ It will be published at a later date. (Enter approximate date. if known).

☐ Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Tom Minnich
[illegible]
per phone call with J. Camp 10/22/98
Authorized Signature/Date

AFRL/FSD
2241 Avionic Circle
Wright -Patterson AFB, OH 45433-7318
John Camp
Print or Type Name

937 255 2164 3512
Telephone Number